**13/08/25 – BACKEND DE APLICACIONES**

**Tipos de variables en Java**

Antes de utilizar una variable, se debe declarar; específicamente unimos tres componentes de esta: el nombre, el tipo de dato y la memoria que esa variable tendrá para almacenar los valores de ella.

**Numéricos**

* Byte: numero entero de un byte, [-128, 127]
* Short: numero entero de 2 bytes, [-32768, 32767]
* Int: numero entero de 4 bytes, [-2^31, 2^31-1]
* Long: numero entero de 8 bytes

**Decimales**

* Float: número decimal simple, precisión 6 o 7 decimales
* Double: número decimal preciso, doble precisión 14 o 15 decimales

**Otros**:

* Char: número sin signo que representa el código de un carácter, usa comillas simples
* Boolean: valor lógico, [true, false]
* String: cadena de caracateres, usa comillas dobles

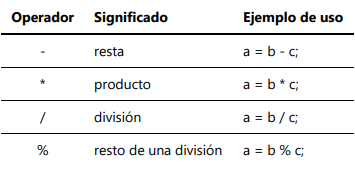
**Restricciones:**

* El nombre o identificador de una variable en Java, solo puede contener letras (mayúsculas y/o minúsculas), o también dígitos (0 al 9), o también el guión bajo ( ` ) (también llamado guión de subrayado).
* El nombre de una variable no debe comenzar con un dígito.
* El nombre de una variable no puede ser una palabra reservada del lenguaje Java.

Java es case sensitive, es decir, hace diferencia entre minúsculas y mayúsculas, por lo que toma como diferentes a dos nombres de variables que no sean exactamente iguales. El identificador contador no es igual al identificador Contador y Java tomará a ambos como dos variables diferentes.

Java utiliza para los nombres de variable (como para casi todo), la notación camel case, es decir los identificadores se definen comenzando con minúscula siempre y se utiliza una sola letra mayúscula para indicar el comienzo de cada nueva palabra en el caso de un nombre compuesto.

**Operadores**

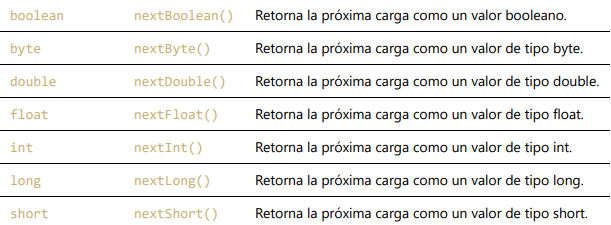


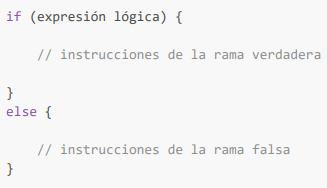
* Si los operadores son byte, short o int, el resultado será int aunque todos los operadores sean byte o short.
* A partir de int java asume el mayor tipo de los operadores involucrados según el siguiente orden: int < long < float < double

**Visualización por pantalla**: System.out.print(). Esta instrucción permite mostrar en pantalla tanto el contenido de una variable como también mensajes formados por cadenas de caracteres.

**Entrada de Datos por teclado en Java**: se utiliza a clase Scanner, la cual entre otras varias y amplias funcionalidades que implementa agrega también la posibilidad de realizar de manera simple y concreta la lectura de valores numéricos, caracteres o cadenas de caracteres desde teclado a través de la consola estándar.

Primero se debe importar: import java.util.Scanner;



**Estructura condicional:** una instrucción condicional contiene una expresión lógica que puede ser evaluada por verdadera o por falsa, y dos bloques de instrucciones adicionales designados en general como la salida o rama verdadera y la salida o rama falsa. La "rama verdadera" se escribe entre llaves, inmediatamente después de cerrar el paréntesis de la condición, y la "rama falsa" va después de la rama verdadera, también envuelta entre llaves, pero precedida de la palabra reservada else.

**Expresiones lógicas, operadores relacionales y conectores lógicos en Java**: una expresión es una fórmula compuesta por variables y constantes (llamados operandos) y por símbolos que indican la aplicación de una acción (llamados operadores).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

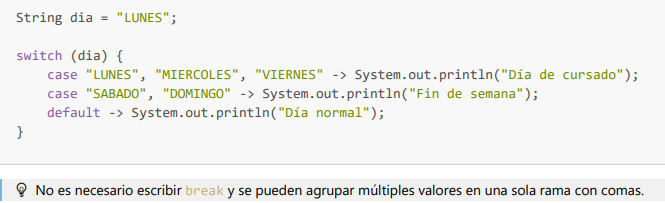
El contenido generado por IA puede ser incorrecto.También es posible emplear operadores conocidos como conectores lógicos para poder chequear varias expresiones lógicas a la vez. Una proposición lógica es una expresión formada por variables y/o constantes relacionadas entre sí mediante operadores de comparación (o relacionales), de tal forma que el resultado de la expresión será un verdadero o un falso.

**Variantes de la expresión condicional en Java**

Condicional Simple: para una condición sólo se especifique la realización de una acción si la respuesta es verdadera y no se requiera hacer nada en caso de responder por falso. Para estos casos, en Java y otros lenguajes es perfectamente válido escribir una instrucción condicional que sólo tenga la rama verdadera, omitiendo por completo la falsa.

Condicional múltiple: cuando es necesario evaluar un valor con más de dos elecciones posibles, se puede resolver con estructuras alternativas anidadas o en cascada, o si se quiere con estructuras simples en secuencia.

Interfaz de usuario gráfica, Texto, Aplicación

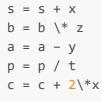
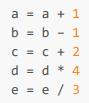
El contenido generado por IA puede ser incorrecto.

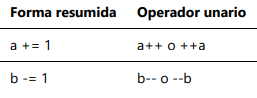
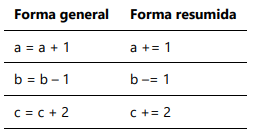
Operador ternario o condicional inline: estructura que permite tomar decisiones basadas en una condición y asignar valores diferentes a una variable en función de si la condición es verdadera o falsa. A diferencia de las estructuras if-else, el operador ternario es más conciso y se utiliza para expresar lógica condicional de manera más compacta.

variable = (condición) ? valor\_verdadero : valor\_falso;



variable = variable operador expresión

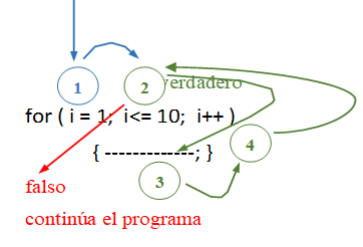




**Estructura repetitiva:**

Existen dos tipos generales de ciclos, los cuales se designan con los siguientes nombres genéricos (el motivo de estos nombres se verá oportunamente en esta misma lección): Ciclos "0 – N" Ciclos "1 – N"

Java implementa los dos tipos de ciclos mediante tres instrucciones específicas, designándolos respectivamente con las palabras reservadas que se marcan a continuación en letra negrita: Ciclo for ( ciclo del tipo "0-N" ), Ciclo while ( ciclo del tipo "0-N" ) y Ciclo do while ( ciclo del tipo "1-N" ).

FOR: se usa en circunstancias en las cuales se conoce de antemano la cantidad exacta de repeticiones que deben realizarse, aunque en el lenguaje Java la sintaxis del ciclo es tan amplia que admite ser usado en cualquier circunstancia que requiera repetición de acciones.

En la cabecera del ciclo existen tres secciones: la primera (i=1 en este caso) se llama sección de inicialización, y se usa para indicar el valor inicial de las variables de control del ciclo. La segunda (i<=10 en este caso) es la condición de control del ciclo: si la condición es verdadera el ciclo ejecuta una repetición del bloque de acciones, pero si es falsa corta el ciclo y continúa el programa. La tercera sección (i++ en este caso) es la

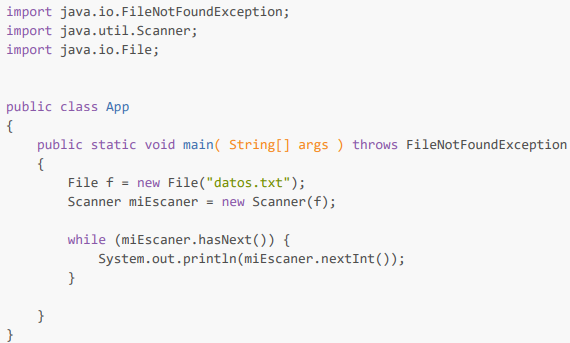
Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

WHILE: necesitamos plantear un ciclo que ejecute en forma repetida un bloque de acciones pero sin conocer previamente la cantidad de vueltas a realizar.

DO WHILE: la condición de control se evalúa por primera vez después de ejecutar por primera vez el bloque de acciones. Esto implica que al menos una vez el bloque de acciones del ciclo siempre será ejecutado, independientemente del valor inicial de la condición de corte (por eso este ciclo es del tipo "1-N": una vez como mínimo se ejecuta el bloque, y luego puede llegar a N repeticiones).

BREAK: El bloque de acciones de un ciclo (while o for) en Java puede incluir una instrucción break para cortar el ciclo de inmediato sin retornar a la cabecera para evaluar la expresión lógica de control o incluir la sentencia continue para volver desde ese punto a la cabecera sin terminar el bloque de sentencias repetitivas.

SCANNER: la clase Scanner es capaz de configurarse con la entrada estándar de java para tomar lo que el usuario carga por teclado sin embargo el uso más común de la clase Scanner es el de la lectura de archivos o flujos de texto.

